Approximation Algorithms for Requirement cut on graphs

Viswanath Nagarajan* R. Ravi†

Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213.

Abstract

In this paper, we unify several graph partitioning problems including multicut, multiway cut, and k-cut, into a single problem. The input to the *requirement cut* problem is an undirected edge-weighted graph $G = (V, E)$, and g groups of vertices $X_1, \cdots, X_g \subseteq V$, with each group X_i having a requirement r_i between 0 and $|X_i|$. The goal is to find a minimum cost set of edges whose removal separates each group X_i into at least r_i disconnected components.

We give an $O(\log n \cdot \log(gR))$ approximation algorithm for the requirement cut problem, where n is the total number of vertices, g is the number of groups, and R is the maximum requirement. We also show that the integrality gap of a natural LP relaxation for this problem is bounded by $O(\log n \cdot \log(gR))$. On trees, we obtain an improved guarantee of $O(\log(gR))$. There is an $\Omega(\log g)$ hardness of approximation for the requirement cut problem, even on trees.

1 Introduction

Graph partitioning problems form a fundamental area of the study of approximation algorithms. The simplest graph partitioning problem is the well known s-t minimum cut problem, where given an edge weighted graph and two specified vertices s and t, the goal is to find a minimum weight set of edges whose removal disconnects s and t. The classical result of Ford & Fulkerson [18] proved a max-flow min-cut duality which related the maximum flow and minimum cut problems.

Multicut: Klein et al. [14] considered a generalization of the s-t minimum cut problem to multiple pairs. In the *multicut* problem, given a set of source-sink pairs $\{(s_1, t_1), \cdots, (s_k, t_k)\}$ in an edge-capacitated graph, the goal is to find a minimum capacity set of edges whose removal separates each s_i from t_i. The corresponding flow problem is *maximum multicommodity flow*: there is one commodity for each s_i-t_i pair, and the objective is to maximize the total flow routed (over all commodities) while respecting the capacities. Klein et al. [14] gave an $O(\log C \cdot \log^2 k)$ approximation algorithm for multicut, where C is the total capacity over all edges. Garg et al. [10] improved the approximation guarantee to $O(\log k)$, which is currently the best known. In [10], the authors also proved that the ratio of the minimum multicut to the maximum multicommodity flow is $\Theta(\log k)$.

Multiway cut: In this problem, there is a set X of terminals, and the goal is to remove a minimum cost set of edges so that no two terminals are in the same connected component. The first

*Supported in part by NSF ITR grant CCR-0122581 (The ALADDIN project) and CCF-0728841. Email: viswa@cmu.edu

†Supported in part by NSF grants CCF-0430751, CCF-0728841 and ITR grant CCR-0122581 (The ALADDIN project). Email: ravi@cmu.edu

approximation algorithm for this problem was due to Dahlhous et al. [7], which gave a guarantee of $2(1-\frac{1}{|X|})$. Using a clever geometric LP relaxation [5], the approximation ratio was improved to 1.3438 in the two papers [5, 11].

Multi-multiway cut: Recently, Avidor & Langberg [4] extended multiway cut and multicut to a multi-multiway cut problem. Given g sets of vertices $X_1, \cdots, X_g$, the goal is to find a minimum cost set of edges whose removal completely disconnects each of the sets $X_1, \cdots, X_g$. The authors [4] presented an $O(\log g)$ approximation algorithm for multi-multiway cut.

Steiner multicut: Another interesting graph partitioning problem is the Steiner multicut problem [12]. In this problem, we are given g groups of vertices $X_1, \cdots, X_g$, and the goal is to find a minimum cost set of edges that separates each group $X_1, \cdots, X_g$. A set S of vertices is said to be separated, if S is not contained in a single connected component. Klein et al. [12] presented an $O(\log^3 gt)$ approximation algorithm for this problem, where $t = \max_{i=1}^{g} |X_i|$ is the maximum size of a group.

k-cut: This is another well studied graph partitioning problem [23], where the goal is to find a minimum cost set of edges that separates the graph into at least k connected components. Saran & Vazirani [23] gave the first approximation algorithm for this problem, which achieves a guarantee of $2 - 2/k$. Alternate algorithms for this problem, achieving the same approximation guarantee were given by [21, 22]. More recently, Chekuri & Guha [6] considered the **Steiner k-cut** problem. This is a generalization of the k-cut problem, where a subset X of vertices is specified as terminals, and the objective is to find a minimum cost set of edges whose removal results in at least k disconnected components, each containing a terminal. Chekuri & Guha [6] showed that the greedy algorithm of [23] can be modified to get a $2 - 2/k$-approximation for this problem. They also showed how to round a natural LP relaxation to achieve the same bound.

In this paper, we study a common generalization that unifies all the graph partitioning problems mentioned above (see Figure 1). The input to the **requirement cut** problem is an n-vertex undirected graph $G = (V, E)$ with non-negative costs c_e on its edges, g groups of vertices $X_1, X_2, \cdots X_g \subseteq V$ with a requirement r_i between 0 and $|X_i|$ for each group X_i. The objective is to find a minimum cost set of edges whose removal separates each group X_i into at least r_i disconnected components (each of which contains at least one member from the group). Below we summarize some of the cut problems in our framework, how requirement cut generalizes them, and the best known approximation results for each of them.

Cut Problem	**Modeling as Requirement cut**	**Best approximation ratio**
Multicut	$\|X_i\| = r_i = 2$ for all $i = 1, \cdots, g$	$O(\log g)$ [10]
Multiway cut	$g = 1$, $r_1 = \|X_1\|$	1.3438 [11]
Multi-multiway cut	$r_i = \|X_i\|$ for all $i = 1, \cdots, g$	$O(\log g)$ [4]
Steiner multicut	$r_i = 2$ for all $i = 1, \cdots, g$	$O(\log^3 gt)$ [12] $O(\log n \cdot \log g)$ (this paper)
Steiner k-cut	$g = 1$, $k = r_1 \leq \|X_1\|$	$2(1 - \frac{1}{k})$ [6]
Requirement cut	—	$O(\log n \cdot \log(gR))$ (this paper)

1.1 Our Results and Paper Outline

We obtain an $O(\log n \cdot \log(gR))$ approximation algorithm for the requirement cut problem on general graphs, where n is the number of vertices in the graph, g is the number of groups and $R = \max_{i=1}^{g} r_i$ is the maximum requirement of any group. We present two algorithms achieving this guarantee.

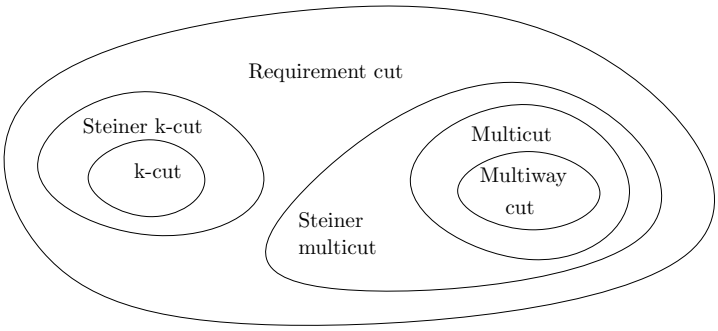

Figure 1: Containment of cut problems

The first (and more interesting) algorithm is via rounding a natural LP relaxation. This also shows that the integrality gap of this LP relaxation is at most $O(\log n \cdot \log(gR))$. The LP rounding procedure is described in Section 2. The second algorithm is based on the greedy heuristic for set cover. The greedy step in this approach is an interesting problem in itself, and has been studied in Klein et al. [12] as the *Steiner ratio cut* problem. We provide an improved approximation algorithm for this problem (when n is at most polynomial in gt), and show that it yields an $O(\log n \cdot \log(gR))$ approximation algorithm for the requirement cut problem. The greedy algorithm is presented in Section 3.

We also show that when restricted to trees, the approximation ratio for requirement cut can be improved to $O(\log(gR))$. On the other hand, a simple reduction from set cover shows that this problem is at least $\Omega(\log g)$ hard to approximate, even on a star (Section 2.2). The LP rounding algorithm on trees generalizes the randomized rounding for set cover, whereas the second method extends the set cover greedy algorithm. While the first algorithm relies on the tree structure for rounding and hence incurs a log-squared overhead, the second method leaves some hope for improvement. The running time of the first algorithm is better, as it involves solving a single linear program.

As noted in the introduction, the Steiner multicut problem of Klein et al. is a special case of the requirement cut problem; so our algorithm implies an improved approximation ratio of $O(\log n \cdot \log g)$ for Steiner multicut. However, we note that using a similar idea even the algorithm in [12] can be shown to achieve this improved guarantee.

2 LP based algorithm

In this section, we present an $O(\log n \cdot \log(gR))$ approximation algorithm for requirement cut based on rounding a natural linear program. We first formulate requirement cut as an integer program, and obtain its linear relaxation (Section 2.1). Then we consider the case when the input graph is a tree, and show that randomized rounding gives an $O(\log(gR))$ approximation (Section 2.2). Finally we show how requirement cut on a general graph can be reduced to requirement cut on a tree through the LP, to obtain an approximation algorithm for the general case (Section 2.3).

2.1 IP formulation and a linear relaxation

We consider an integer program for the requirement cut problem, and a linear relaxation for it. This formulation is a generalization of that used in Chekuri & Guha [6] for the Steiner k-cut problem. By adding edges of zero cost, we may assume that the input graph $G = (V, E)$ is complete. Our IP has a 0-1 variable d_e for each edge $e \in E$, which represents whether or not this edge is cut.

$$
\begin{array}{lll}
 & \min \sum_{e \in E} c_e d_e & \\
 & s.t. & \\
(Steiner - IP) & \sum_{e \in T_i} d_e \geq r_i - 1 & \forall T_i : \text{Steiner tree on } X_i, \quad \forall i = 1 \cdots g \\
 & d_e \in \{0, 1\} & \forall e \in E
\end{array}
$$

This integer program is clearly an exact formulation of the requirement cut problem. However, the LP relaxation of $Steiner - IP$ does not have a polynomial time separation oracle (the separation problem is minimum Steiner tree). So we consider a relaxation of the Steiner tree constraints, by requiring that all *spanning trees* on the induced graph $G[X_i]$ have length at least $r_i - 1$, for each group X_i. Observe that, in any minimal solution to $(Steiner - IP)$, the edge variables d satisfy the triangle inequality: i.e. for any vertices $u, v, w \in V$, if $d_{(u,v)} = d_{(v,w)} = 0$ then $d_{(u,w)} = 0$. So addition of the triangle inequality constraints to $(Steiner - IP)$ does not change the optimal solution. Relaxing the integrality of d, we obtain the following linear programming relaxation for requirement cut.

$$
\begin{array}{lll}
 & \min \sum_{e \in E} c_e d_e & \\
 & s.t. & \\
(LP) & \sum_{e \in T_i} d_e \geq r_i - 1 & \forall T_i : \text{spanning tree in } G[X_i], \quad \forall i = 1 \cdots g \\
 & d_{(u,w)} \leq d_{(u,v)} + d_{(v,w)} & \forall u, v, w \in V \\
 & 0 \leq d_e \leq 1 & \forall e \in E
\end{array}
$$

Note that (LP) can be solved in polynomial time using the ellipsoid algorithm (using a minimum spanning tree algorithm in the separation oracle). Let d^* denote an optimal solution to (LP), and define a new length function d as $d_e = \min\{2 \cdot d^*_e, 1\}$ for all $e \in E$. Since d^* is a metric, so is d. It is also clear that edge lengths in both d^* and d are in $[0, 1]$. The next claim follows from the MST heuristic for Steiner tree.

Claim 1 *For any group X_i $(i = 1, \cdots g)$, the minimum Steiner tree on X_i w.r.t. d has length at least $r_i - 1$.*

Proof: We fix a group i for the rest of the proof. Let $S = (V(S), E(S))$ be the minimum Steiner tree (under metric d) on group X_i. Denote the length of S by $d(S)$. We will construct a *spanning tree* S' on X_i which has d^*-length $d^*(S') \leq d(S)$. Since d^* is a feasible solution to the linear program (LP), the claim would follow. Vertices in X_i are referred to as terminals, and vertices in $V \setminus X_i$ are Steiner vertices. Edges in which *both* end points are terminals are called terminal edges, and all other edges are Steiner edges.

We first modify S so that the only length 1 (in metric d) edges in S are terminal edges. If (u, v) is an edge in S with $d_{u,v} = 1$ and u is a Steiner vertex, then consider removing edge (u, v) from S to obtain 2 subtrees S_u and S_v. Clearly there is at least one terminal in each of S_u and S_v. Arbitrarily add an edge (u', v') to S where $u' \in S_u$ and $v' \in S_v$ are terminals. It is clear that the length of S does not increase since $d_{u',v'} \leq 1$. So we may assume that all Steiner edges in S have

d-length strictly less than 1. Now consider the length function d^*: from the preceding argument, any Steiner edge $e \in E(S)$ has length $d^*_e = d_e/2$.

We now shortcut over Steiner vertices in S to obtain a spanning tree S' on X_i, as follows. Let $E_t \subseteq E(S)$ denote the set of terminal edges in S, and $\{T_j\}_{j=1}^{l}$ the trees in the forest induced by $E(S) \setminus E_t$. Note that in each tree T_j, all leaves are terminals and all edges are Steiner edges. Taking an Euler tour of each tree T_j, we obtain tree T'_j over just the terminals spanned by T_j; since all edges of T_j are Steiner edges, $d^*(T'_j) \leq 2 \cdot d^*(T_j) = d(T_j)$. We now obtain the spanning tree S' on X_i as $S' = E_t \cup (\cup_{j=1}^{l} T'_j)$. Since $d^*_e \leq d_e$ for all edges, we can bound the length of S' as $d^*(S') = \sum_{e \in E_t} d^*_e + \sum_{j=1}^{l} d^*(T'_j) \leq \sum_{e \in E_t} d^*_e + \sum_{j=1}^{l} d(T_j) \leq \sum_{e \in E_t} d_e + \sum_{e \in E(S) \setminus E_t} d_e = d(S)$. This proves the claim.■

2.2 LP rounding for requirement cut on trees

In this section, we consider a special case of requirement cut when the input graph is a tree. We show how to round the linear program (LP) within a factor of $O(\log gR)$, to obtain an approximation algorithm for requirement cut on trees. We note that even this restriction is at least as hard to approximate as set-cover. Consider a star with one edge corresponding to each set in the set-cover instance. For each element j, we create a group that contains the root, and the leaves of all edges corresponding to sets containing j. Further, we set the requirement of each group to 2, and all edge costs to 1. Clearly, there is a one-to-one correspondence of feasible solutions to the requirement cut instance and the set cover instance, which also preserves the cost. This shows that requirement cut on trees is at least $\Omega(\log g)$ hard to approximate [9].

Given a requirement cut instance on a tree $T = (V, E)$, the algorithm begins by solving (LP) optimally and obtaining the solution d defined in Claim 1. Let OPT^* denote the optimal value of (LP). We now describe how d is rounded to an integral solution. Our algorithm proceeds in phases, and augments the (partial) solution in each phase. Let $C^k \subseteq E$ denote the partial solution at the start of the phase k; so $C^1 = \phi$. Let $F^k = T \setminus C^k$ denote the forest in phase k, with the edges in C^k removed. Each phase involves a randomized rounding for all the edges: the rounding in phase k picks each edge $e \in F^k$ independently with probability d_e, and adds all chosen edges to the partial solution C^k to get C^{k+1}. It is clear that the expected cost in each phase is at most $\sum_{e \in F} c_e d_e \leq 2 \sum_{e \in E} c_e d^*_e = 2 \cdot OPT^*$.

Let c^k_i denote the number of connected components containing X_i in F^k. The *residual requirement* of group X_i at the start of phase k is defined to be $s^k_i = \max\{0, r_i - c^k_i\}$. The rounding procedure ends when the residual requirement of each group is 0, and the requirement of each group is completely satisfied at this point. We show that the expected number of phases in this algorithm is $O(\log gR)$, which gives us the desired approximation guarantee. The main step in this is to show that in each phase, the total residual requirement (summed over all groups) goes down by a constant factor in expectation. This technique was also used in the paper of Konjevod et al. [15] on the covering Steiner problem.

2.2.1 Rounding in a single phase

The analysis here is for a single phase, and we drop the superscript k for ease of notation. For a group X_i, define F_i to be the sub-forest induced by X_i in F. Let H_i be the forest obtained from F_i by short cutting over all degree two Steiner (non X_i) vertices. So all Steiner vertices in H_i have degree at least 3. Such a forest is useful because of the following claim.

Claim 2 *Suppose $H = (V(H), E(H))$ is a forest with vertices $X \subseteq V(H)$ denoted terminals, and with each vertex $V(H) \setminus X$ having degree at least 3. Then, the removal of* any *$m \geq 1$ edges of $E(H)$ results in at least $\lceil \frac{m+1}{2} \rceil$ more components containing terminals.*

Proof: It suffices to prove the claim when H is a tree. The forest case can be obtained by adding the contributions from its trees (and using the fact that $\lceil a \rceil + \lceil b \rceil \geq \lceil a + b \rceil$). Consider any set $A \subseteq E(H)$ of $m \geq 1$ edges, and the components $C_1, C_2, \cdots C_{m+1}$ of $H \setminus A$. Denote a component C_i to be terminal, if it contains a terminal, and non-terminal otherwise. We can think of A as a tree T' on the node set $\{C_1, \cdots C_{m+1}\}$. Suppose that a non-terminal component C_i has l non terminals, and f edges leaving it. Since each non-terminal has degree ≥ 3, the total degree in C_i is at least $3l$. There are exactly $l - 1$ internal edges in C_i, so we have a total degree $2l - 2 + f \geq 3l$, *i.e.* $f \geq l + 2 \geq 3$. Thus, if we look at the tree T', each non terminal component has degree at least 3. In such a tree, there are at least $\lceil \frac{|T'|}{2} \rceil + 1 = \lceil \frac{m+1}{2} \rceil + 1$ terminal C_is, *i.e.* $H \setminus A$ has at least $\lceil \frac{m+1}{2} \rceil$ more terminal components.■

Recall that s_i is the residual requirement of group X_i at the start of the current phase. For any subgraph F' of F, the length of F' is $d(F') = \sum_{e \in F'} d_e$. For any pair of vertices $u, v \in V$, let $p_{u,v}$ denote the *probability* that vertices u and v are disconnected in this phase of rounding.

Lemma 1 *The total probability weight on H_i, $\sum_{e \in H_i} p_e$, is at least $(1 - \frac{1}{e}) \cdot d(H_i)$.*

Proof: For those edges e of H_i which are also edges in F, it is clear that $p_e = d_e$. Now consider an edge $(u, v) \in H_i$ that is obtained by short cutting a path P in F_i. We claim that $p_{u,v} \geq (1 - \frac{1}{e}) d_{u,v}$. Since each edge is rounded independently, and u and v are separated if any of the edges in P is removed, $p_{u,v} = 1 - \Pi_{e \in P}(1 - d_e) \geq 1 - e^{-d(P)}$. We consider the following 2 cases:

- $d(P) \leq 1$. Note that $1 - e^{-y} \geq (1 - 1/e)y$ for $y \in [0, 1]$. So in this case, $p_{u,v} \geq (1 - 1/e) d(P) \geq (1 - 1/e) d_{u,v}$, since d is a metric.
- $d(P) \geq 1$. In this case $p_{u,v} \geq 1 - 1/e \geq (1 - 1/e) d_{u,v}$, since $d_{u,v} \in [0, 1]$.

Thus, summing $p_{u,v}$ over all edges $(u, v) \in H_i$ we get the lemma. ■

Now consider adding edges to forest H_i to make it a *Steiner tree* on X_i. If X_i appears in c_i connected components in F, we need to add $c_i - 1$ edges. Since every edge has d-length at most 1, adding $c_i - 1$ edges increases the length of H_i by at most $c_i - 1$. But from Claim 1, every Steiner tree on X_i has length at least $r_i - 1$. So we get $d(H_i) + (c_i - 1) \geq (r_i - 1)$, *i.e.*, $d(H_i) \geq r_i - c_i = s_i$ (assuming that group X_i has residual requirement $s_i \geq 1$). Lemma 1 then implies that $\sum_{e \in H_i} p_e \geq (1 - 1/e) s_i \geq \frac{s_i}{2}$.

Consider a 0-1 random variable Z_e^i for each edge $e = (u, v) \in H_i$, which is 1 iff vertices u and v are disconnected in this phase, and 0 otherwise; note that $Pr[Z_e^i = 1] = p_e$. The edges in forest F corresponding to each $e \in H_i$ are disjoint; so the random variables Z_e^i (for $e \in H_i$) are independent. Let $Y_i = \sum_{e \in H_i} Z_e^i$, which is the number of edges cut in forest H_i. Although H_i is not a subgraph of F, disconnecting vertices in H_i is equivalent to disconnecting them in F. Now, $E[Y_i] = \sum_{e \in H_i} E[Z_e^i] = \sum_{e \in H_i} p_e \geq \frac{s_i}{2}$. We make use of the following version of the Chernoff bound (see eg., Motwani & Raghavan [19], Theorem 4.2). [1]

[1] In the preliminary version of this paper [20], we used only linearity of expectation in the analysis, which is incorrect. This stronger deviation bound is required for the rounding analysis to work.

Lemma 2 (Chernoff bound) *Let $I_1, \cdots, I_n$ be independent 0-1 random variables, $I = \sum_{j=1}^{n} I_j$, and $E[I] = \mu$. Then for any $0 < \delta < 1$, $Pr[I < (1-\delta)\cdot\mu] < e^{-\mu\delta^2/2}$.*

Since Y_i is the sum of independent 0-1 random variables Z_e^i, and $E[Y_i] \geq \frac{s_i}{2}$, Lemma 2 implies the following for any group X_i with positive residual requirement ($s_i \geq 1$):

$$Pr[Y_i < \frac{s_i}{4}] \leq Pr[Y_i < \frac{1}{2}E[Y_i]] < e^{-E[Y_i]/8} \leq e^{-s_i/16} \leq e^{-1/16}$$

Let N_i be the increase in the number of components of group X_i in this phase. Since H_i is a forest that satisfies the conditions of Claim 2, we have $N_i \geq Y_i/2$. Thus we have:

$$Pr[N_i < \frac{s_i}{8}] \leq Pr[Y_i < \frac{s_i}{4}] \leq e^{-1/16} \tag{1}$$

2.2.2 Bounding the number of phases

Let random variable S_i^k denote the residual requirement of group X_i at the *start* of phase k, and N_i^k the increase in the number of components of group X_i in phase k. Note that $S_i^{k+1} = \max\{S_i^k - N_i^k, 0\}$. The analysis in Section 2.2.1 holds for any phase k. Rewriting inequality (1), we have $q = Pr[S_i^{k+1} > \frac{7}{8}s_i | S_i^k = s_i] = Pr[N_i^k < \frac{s_i}{8} | S_i^k = s_i] \leq e^{-1/16}$ (although (1) requires $s_i \geq 1$, note that this inequality is trivial when $s_i = 0$). Thus,

$$E[S_i^{k+1} | S_i^k = s_i] \leq s_i \cdot q + \frac{7s_i}{8} \cdot (1-q) = \frac{7+q}{8}s_i$$

So unconditionally, $E[S_i^{k+1}] \leq \alpha \cdot E[S_i^k]$, where $\alpha = \frac{7+q}{8} \leq \frac{7+e^{-1/16}}{8}$ is a constant less than 1. Now let $S^k = \sum_{i=1}^{g} S_i^k$ be the total residual requirement at the start of phase k. By linearity of expectation, $E[S^{k+1}] = \sum_{i=1}^{g} E[S_i^{k+1}] \leq \alpha \sum_{i=1}^{g} E[S_i^k] = \alpha \cdot E[S^k]$. Applying this inequality recursively, we have that after k phases, $E[S^{k+1}] \leq \alpha^k E[S^1] \leq \alpha^k \cdot gR$, since the total residual requirement at the start of the algorithm $S^1 = \sum_{i=1}^{g} r_i \leq gR$. So if we choose $k^* = \frac{\log_2(gR)+2}{\log_2(1/\alpha)} = O(\log(gR))$, $E[S^{k^*+1}] \leq 1/4$. Using the Markov inequality, $Pr[S^{k^*+1} \geq 1] \leq 1/4$. Recall that the expected cost in each phase is at most $2 \cdot OPT^*$; so after k^* phases the expected total cost is at most $2k^* \cdot OPT^*$. Again applying the Markov inequality, with probability at least $\frac{3}{4}$, the total cost after k^* phases is at most $8k^* \cdot OPT^*$. So with probability at least $\frac{1}{2}$, this rounding algorithm produces a feasible solution of cost at most $8k^* \cdot OPT^* = O(\log(gR)) \cdot OPT^*$. Thus, we have proved the following.

Theorem 1 *There is a polynomial time randomized rounding algorithm for requirement cut on trees, that obtains a solution of cost $O(\log(gR))$ times the optimal value of (LP).*

This also shows that the integrality gap of (LP) on trees is $O(\log(gR))$. A lower bound of $\Omega(\log g)$ on the integrality gap of (LP) follows from the integrality gap example for the set cover linear program. So the analysis in this section is almost tight.

Remark: An alternate rounding algorithm. The rounding algorithm that was analyzed above can be summarized as follows: if there are k^* phases, the final (randomized) solution S is obtained by including each edge e of tree T independently with probability $q_e = 1 - (1-d_e)^{k^*}$. The proof of Theorem 1 implies that when $k^* \geq c \cdot \log(gR)$ (for a sufficiently large constant c), the solution S is feasible to the requirement cut instance with probability at least $\frac{3}{4}$.

A related rounding procedure involves picking each edge e of T independently with probability $\tilde{q}_e = \min\{k^* \cdot d_e, 1\}$ (for a suitable value of k^*); let $\tilde{S}$ denote the resulting randomized solution. Observe that for any $e \in T$, $q_e \leq 1$ and $q_e = 1 - (1 - d_e)^{k^*} \leq k^* \cdot d_e$. Thus $q_e \leq \tilde{q}_e$ for all $e \in T$; in other words, the solution $\tilde{S}$ stochastically dominates S. We use the following *monotone property* of the requirement cut problem: if S_1 is a feasible solution to an instance of requirement cut and $S_2 \supseteq S_1$, then S_2 is also a feasible solution. When $k^* \geq c \cdot \log(gR)$ (for a large constant c), solution S is feasible to the requirement cut instance with probability at least $\frac{3}{4}$ (as mentioned above); combined with the monotone property and the fact that $\tilde{S}$ dominates S, it follows that $\tilde{S}$ is also a feasible solution with probability at least $\frac{3}{4}$. As in the proof of Theorem 1, since the expected cost of solution $\tilde{S}$ is $k^* \cdot OPT^*$, $\tilde{S}$ is a feasible solution of cost at most $8k^* \cdot OPT^*$ with probability at least $\frac{1}{2}$.

2.3 LP rounding for requirement cut

In this section we show how the linear program (LP) yields an $O(\log n \cdot \log(gR))$ approximation algorithm for requirement cut on general graphs. This would also show that the integrality gap of (LP) is at most $O(\log n \cdot \log(gR))$. Let $\mathcal{I}$ denote any instance of requirement cut on a graph $G = (V(G), E(G))$, with groups $X_1, \cdots, X_g$ and corresponding requirements $r_1, \cdots, r_g$. Let LP_G denote the LP relaxation for $\mathcal{I}$, d^* its optimal solution, and OPT^* its optimal value. Our rounding procedure uses solution d^* (which defines a metric), and embeds it as a fractional solution into a distribution of tree instances. Then we use the rounding algorithm for requirement cut on trees (Section 2.2) to obtain an integral solution on a tree instance, which also corresponds to a solution to the requirement cut instance $\mathcal{I}$. We use the following embedding result of Fakcharoenphol et al. [8].

Theorem 2 ([8]) *For any metric (V, d) with $|V| = n$, there exists a distribution $\mathcal{T}$ of tree metrics satisfying:*

1. *For all $(T, \kappa) \in \mathcal{T}$, $\kappa_{i,j} \geq d_{i,j}$, $\forall\ i, j \in V$.*
2. *$E_{\mathcal{T}}[\kappa_{i,j}] \leq \rho \cdot d_{i,j}$, $\forall\ i, j \in V$, where $\rho = O(\log n)$.*

Furthermore, trees from distribution $\mathcal{T}$ can be sampled in polynomial time.

The rounding algorithm on general graphs first embeds metric $(V(G), d^*)$ into a distribution $\mathcal{T}$ of dominating tree metrics, as in Theorem 2. Let $(T, \kappa) \in_R \mathcal{T}$ be a random tree metric from this distribution. Here $T = (V(T), E(T))$ is a tree on the vertex set $V(G)$ plus some additional (Steiner) vertices, and κ defines a tree metric on $V(T)$ by assigning distances to the tree edges $E(T)$. For any edge $e \in E(T)$, we denote by sep_e the set of edges of the original graph G that are separated by e: $sep_e = \{(i,j) | i, j \in V(G), e \text{ lies on the } i - j \text{ path in } T\}$. Edge costs in T are defined as follows: $c'_e = \sum_{(i,j) \in sep_e} c_{i,j}$ for all $e \in E(T)$ (non tree edges have cost 0). Consider a random instance $\mathcal{J}$ of requirement cut on trees, defined on T with cost function c', and groups $X_1, \cdots, X_g$ having requirements $r_1, \cdots, r_g$ (same as in $\mathcal{I}$). Given any feasible integral solution S to the tree instance $\mathcal{J}$, it is clear that the edge-set $\cup_{e \in S} sep_e$ defines a feasible integral solution to $\mathcal{I}$ of the *same* cost.

Now consider the LP relaxation LP_T of $\mathcal{J}$ (a requirement cut instance on trees). Define $\kappa'_{u,v} = \min\{\kappa_{u,v}, 1\}$ for all $u, v \in V(T)$; note that κ' is a metric with distances in $[0, 1]$. Since κ restricted to $V(G)$ dominates d^* (Theorem 2) and d^* is a metric with distances in $[0, 1]$, κ' restricted to $V(G)$

also dominates d^*. So for any spanning tree T_i on a group X_i, its length under κ', $\kappa'(T_i) \geq d^*(T_i)$, its length under d^*. Since d^* is a feasible solution for LP_G, we get $\kappa'(T_i) \geq r_i - 1$. Thus κ' is a feasible (fractional) solution to LP_T. The cost of this fractional solution is given by the random variable $C = \sum_{u,v \in V(T)} c'_{u,v} \cdot \kappa'_{u,v} = \sum_{e \in E(T)} c'_e \cdot \kappa'_e \leq \sum_{e \in E(T)} c'_e \cdot \kappa_e$ (since κ dominates κ'). Now since κ is a tree metric,

$$C \leq \sum_{e \in E(T)} c'_e \cdot \kappa_e = \sum_{e \in E(T)} \kappa_e \sum_{(i,j) \in sep_e} c_{i,j} = \sum_{i,j \in V(G)} c_{i,j} \sum_{e:(i,j) \in sep_e} \kappa_e = \sum_{i,j \in V(G)} c_{i,j} \cdot \kappa_{i,j}$$

Using Theorem 2 and linearity of expectation, we get $E[C] \leq \rho \cdot \sum_{i,j \in V(G)} c_{i,j} d^*_{i,j} = O(\log n) \cdot OPT^*$. Now using the rounding algorithm for trees (Theorem 1), we get an integral solution to $\mathcal{J}$ of expected cost at most $O(\log n \cdot \log(gR))$. Since any integral solution to $\mathcal{J}$ also corresponds to an integral solution to $\mathcal{I}$, we obtain the following.

Theorem 3 *There is a polynomial time randomized rounding algorithm for requirement cut on graphs, that obtains a solution of cost $O(\log n \cdot \log(gR))$ times the optimal value of (LP).*

Remark: Improved approximation for small sized groups. We note that there is an alternate algorithm that gives an $O(t \log g)$ approximation ratio, where $t = \max_{i=1}^{g} |X_i|$ is the size of the largest group. This follows from the algorithm for multi-multiway cut [4]. We first solve the LP relaxation (LP) to get a metric d^* with objective value OPT^*. The following argument holds for any group X_i. Consider a minimum spanning tree T_i on X_i, with length $d^*(T_i) \geq r_i - 1$. Since each edge has length at most 1, T_i has at least $r_i - 1$ edges of length at least $\frac{1}{t}$. Removing the $r_i - 1$ longest edges in T_i, we obtain r_i connected components. Pick one X_i-vertex from each component to obtain set $S_i = \{s_1, s_2, \cdots, s_{r_i}\} \subseteq X_i$. Since T_i is an MST on X_i, each pairwise distance (under metric d^*)in S_i is at least $\frac{1}{t}$.

Define a new metric $d' = \min\{t \cdot d^*, 1\}$. In metric d', the distance between any pair of vertices in S_i is 1 (for each $i = 1, \cdots, g$). Consider a multi-multiway cut instance on the sets $S_1, S_2, \cdots, S_g$: recall that the goal is to remove a minimum cost set of edges that completely disconnects each of $\{S_i\}_{i=1}^{g}$. The approximation algorithm in Avidor and Langberg [4] was based on the following LP relaxation for multi-multiway cut.

$$\begin{array}{lll} \min \sum_{e \in E} c_e d_e & & \\ s.t. & & \\ d_{(x,y)} \geq 1 & \forall x, y \in S_i, & \forall i = 1 \cdots g \\ d_{(u,w)} \leq d_{(u,v)} + d_{(v,w)} & \forall u, v, w \in V & \\ 0 \leq d_e \leq 1 & \forall e \in E & \end{array}$$

Observe that d' is a feasible fractional solution to this linear program. So using the algorithm in [4], we obtain a set of edges whose removal completely disconnects each S_i, and has cost at most $O(\log g)(d' \cdot c) \leq O(t \cdot \log g) \cdot OPT^*$. It is clear that this solution is also feasible to the original requirement cut instance. Thus we have an $O(t \cdot \log g)$ approximation for requirement cut, which is an improvement when the largest group size t is a constant.

3 Greedy algorithm

In this section we present a greedy algorithm for the requirement cut problem that achieves a guarantee of $O(\log n \cdot \log(gR))$. This algorithm is deterministic, and its approximation ratio matches

that of the randomized rounding algorithm (Section 2). The greedy algorithm works in phases, and maintains a partial solution in every phase. Each phase is a greedy step that augments the partial solution. The algorithm ends when the requirements of all groups have been satisfied. We first define an appropriate greedy subproblem to solve in each phase, then obtain an approximation algorithm for this subproblem (Section 3.1), and finally show how this can be used to solve the requirement cut problem (Section 3.2).

Consider an instance of requirement cut on graph $G = (V, E)$, and groups $X_1, \cdots, X_g$ having requirements $r_1, \cdots, r_g$. As before, by adding 0 cost edges, we assume that graph G is complete. Let OPT^* denote the optimal cost of this instance. The partial solution at the start of phase k is a set of edges $C^k \subseteq E$, and the *residual graph* in phase k is $G^k = G \setminus C^k$. For any group X_i, the *residual requirement* in phase k is the number of additional components that X_i should be split into, in order to satisfy its requirement (see also Section 2.2). The total residual requirement (over all groups) in phase k is denoted ϕ_k. In any phase, a group X_i is said to be *active* if it has a positive residual requirement. At the start of the algorithm, $C^1 = \phi$ and the residual requirement of each group X_i is $r_i - 1$.

In phase k, a *cut* in graph $G^k = (V, E \setminus C^k)$ refers to a set of edges of the form $\partial_k S \doteq \{(i,j) \in E \setminus C^k | i \in S, j \notin S\}$, where $S \subseteq V$ is a set of vertices. The *coverage* of cut $\partial_k S$, $cov(\partial_k S)$, is defined as the *number* of active groups (in phase k) that are separated by $\partial_k S$ in G^k. Note that this definition of coverage does not take into account, the increase in the number of components of a group: this is because if the increase is more than the residual requirement of the group, it does *not* help in satisfying any requirement. The *cost-effectiveness* of cut $\partial_k S$ is defined to be the ratio of its cost to its coverage, and is denoted $eff(\partial_k S) = \frac{c(\partial_k S)}{cov(\partial_k S)}$. In each phase k, the greedy algorithm adds a cut of (approximately) minimum cost effectiveness to its partial solution, to obtain C^{k+1}. Since the algorithm only removes cuts and the initial graph G is complete, in each phase, all the connected components are complete [2]. The next lemma shows why the minimum cost-effective cut is a good choice as a greedy step.

Lemma 3 *In any phase k, there is a cut $\partial_k M$ such that M is contained in some connected component of G^k, and $eff(\partial_k M) \leq \frac{2OPT^*}{\phi_k}$.*

Proof: Let $S_1, S_2, \cdots S_l$ denote the connected components in the graph obtained by removing the edges of an optimal solution S^* from the residual graph G^k. Consider the cuts $\{\partial_k S_i\}_{i=1}^l$. It is clear that each edge in $\cup_{i=1}^l \partial_k S_i$ is contained in S^*, and participates in exactly two cuts in $\{\partial_k S_i\}_{i=1}^l$. So we have $\sum_{i=1}^l c(\partial_k S_i) \leq 2 \cdot OPT^*$.

Starting with graph G^k, consider removing cuts step by step, in the order $\partial_k S_1, \partial_k S_2, \cdots \partial_k S_l$, to end up with the graph $G^k \setminus S^*$ having connected components $\{S_i\}_{i=1}^l$. Note that all connected components in each step of this procedure are complete, and the cut removed in each step is contained in some connected component. So in any single step above, the increase in the number of connected components is at most 1; and the additional requirement satisfied for any group in one step is also at most 1. So the total (over all groups) additional requirement satisfied in any one step is *equal* to the number of (currently active) groups separated in that step. Clearly any group that is active at some step in this procedure is also active in G^k (*i.e.*, in phase k). So the total additional requirement satisfied over all groups and all steps is at most $\sum_{i=1}^l cov(\partial_k S_i)$. On the other hand, since S^* is a feasible solution to the requirement cut instance, the total residual

[2] A connected component consisting of vertices $U \subseteq V$ is said to be complete if every edge with both end-points in U is present.

requirement in $G^k \setminus S^*$ is 0. But the total residual requirement in G^k is ϕ_k; so the the total additional requirement satisfied (over all groups & steps) in this procedure is at least ϕ_k. Thus we have $\sum_{i=1}^{l} cov(\partial_k S_i) \geq \phi_k$. Now,

$$\min_{i=1}^{l} \ eff(\partial_k S_i) = \min_{i=1}^{l} \frac{c(\partial_k S_i)}{cov(\partial_k S_i)} \leq \frac{\sum_{i=1}^{l} c(\partial_k S_i)}{\sum_{i=1}^{l} cov(\partial_k S_i)} \leq 2\frac{OPT^*}{\phi_k}$$

So the minimum cost effective cut amongst $\{\partial_k S_i\}_{i=1}^{l}$ has cost-effectiveness at most $\frac{2OPT^*}{\phi_k}$. Furthermore, it is clear that each $\{S_i\}_{i=1}^{l}$ lies in some connected component of G^k.■

3.1 The Steiner ratio problem

In making the greedy choice in any phase, we wish to compute a cut of minimum cost effectiveness. Formally, we are given a connected graph $H = (V(H), E(H))$ with costs c_e on edges, and g groups of vertices $X_1, \cdots, X_g \subseteq V(H)$. Recall that a cut in graph H is any set of edges of the form $\partial S = \{(i,j) \in E(H) : i \in S,\ j \notin S\}$, where $S \subseteq V(H)$. The goal is to find a cut that minimizes the ratio of its cost to the number of groups that it separates. This is also the *minimum ratio Steiner cut* problem that was studied in Klein et al. [12], where an $O(\log^2 gt)$ approximation algorithm was given. Here, using the results of Fakcharoenphol et al. [8], we improve the approximation ratio to $O(\log n)$. We note that this is an improvement over [12] when n is at most polynomial in gt; otherwise the earlier $O(\log^2 gt)$ guarantee is better. When all groups have size 2, the Steiner ratio problem reduces to sparsest cut [16, 3, 17, 2], for which the currently best known approximation guarantee is $O(\sqrt{\log g} \cdot \log\log g)$ due to Arora et al. [1].

The approximation algorithm for the Steiner ratio problem is based on rounding a natural linear programming relaxation, which is described below. This is also the LP relaxation that was used in Klein et al. [12]. Using the MST algorithm in the separation oracle, this LP can be solved by the Ellipsoid algorithm in polynomial time. Again, we assume that the graph H is complete, by adding edges of zero cost.

$$(LP - ratio) \qquad \begin{array}{ll} \min \sum_{e \in E(H)} c_e l_e & \\ s.t. & \\ \sum_{i=1}^{g} \sum_{e \in T_i} l_e \geq 1 & \forall\ (T_1, \cdots, T_g) \text{ where,} \\ & T_i : \text{spanning tree on } H[X_i],\ \forall i = 1, \cdots, g \\ 0 \leq l_e \leq 1 & \forall e \in E(H) \\ l_{(u,w)} \leq l_{(u,v)} + l_{(v,w)} & \forall u, v, w \in V(H) \end{array}$$

To see that this is indeed a relaxation of the Steiner ratio problem, consider the optimal solution B^* (which is a cut) of the Steiner ratio problem. Define a 0-1 metric l' corresponding to B^* by setting $l'(u,v) = 0$ iff u and v are in the same connected component in $H \setminus \partial B^*$, and 1 otherwise. Let σ be the number of groups separated by ∂B^*. Clearly, the sum of the minimum spanning trees in $H[X_i]$ (for $i = 1, \cdots g$) is also σ. Now the metric $\frac{1}{\sigma} \cdot l'$ is a feasible solution to $(LP - ratio)$, and has cost $\frac{c(\partial B^*)}{\sigma} = eff(\partial B^*)$, which is the optimal value of the Steiner ratio problem.

Let l be the optimal solution to $(LP - ratio)$, and $OPT' = \sum_{e \in E(H)} c_e \cdot l_e$ its cost. We now describe how l can be rounded to get an approximately optimal cut. Using Theorem 2, the algorithm first embeds metric $(V(H), l)$ to a distribution $\mathcal{T}$ of dominating tree metrics. Let $(T, \kappa) \in_R \mathcal{T}$ be a random sample from $\mathcal{T}$. As defined in Section 2.3, each edge $f \in T$ corresponds to a cut in graph

H, $sep_f = \{(i,j) | i,j \in V(H), f \text{ lies on the } i-j \text{ path in } T\}$. We say that edge $f \in T$ separates group i if removing edge f from T disconnects X_i, and denote this $f|X_i$. Below, $T[X_i]$ denotes the tree induced by X_i in T, and MST_i denotes the minimum spanning tree on X_i in the specified metric. We have,

$$\begin{aligned}
\frac{c(sep_f)}{cov(sep_f)} &= \min_{f \in T} \left[\frac{\kappa_f c(sep_f)}{\kappa_f \sum_{i=1}^{g} 1_{f|X_i}} \right] && \leq \frac{\sum_{f \in T} \kappa_f \sum_{(i,j) \in sep_f} c_{i,j}}{\sum_{f \in T} \kappa_f \sum_{i=1}^{g} 1_{f|X_i}} \\
&= \frac{\sum_{i,j} c_{i,j} \sum_{f:(i,j) \in sep_f} \kappa_f}{\sum_{f \in T} \kappa_f \sum_{i=1}^{g} 1_{f|X_i}} && = \frac{\sum_{i,j} c_{i,j} \kappa_{i,j}}{\sum_{i=1}^{g} \sum_{f:f|X_i} \kappa_f} \\
&= \frac{\sum_{i,j} c_{i,j} \kappa_{i,j}}{\sum_{i=1}^{g} \kappa(T[X_i])} && \leq 2 \cdot \frac{\sum_{i,j} c_{i,j} \kappa_{i,j}}{\sum_{i=1}^{g} \kappa(MST_i)} \qquad (1) \\
&\leq 2 \cdot \frac{\sum_{i,j} c_{i,j} \kappa_{i,j}}{\sum_{i=1}^{g} l(MST_i)} && \qquad (2) \\
&\leq 2 \cdot \sum_{i,j} c_{i,j} \kappa_{i,j} && \qquad (3)
\end{aligned}$$

The only non trivial inequalities above are the last three. Since $T[X_i]$ is a Steiner tree on X_i, the length of an MST on X_i is at most 2 times the length of $T[X_i]$, so (1) follows. For (2), note that κ restricted to $V(H)$ dominates l, and (3) follows from the feasibility of l in $(LP-ratio)$. It was shown in [8] that, given a metric (in this case, l) and weights on all pairs of vertices (in this case $c_{i,j}$), their tree embedding algorithm can be derandomized to find a *single tree* with weighted average stretch at most $O(\log n)$ times that of metric l. Thus we can find (in deterministic polynomial time), a single tree metric (T', κ') such that $\sum_{i,j} c_{i,j} \cdot \kappa'_{i,j} \leq \rho \cdot \sum_{i,j} c_{i,j} l_{i,j}$, where $\rho = O(\log n)$. Given such a tree, the algorithm outputs the minimum cost effective cut B corresponding to an edge of T', as the approximate solution. From the above argument, it follows that $eff(\partial B) \leq 2\rho \cdot OPT'$, and we obtain an $O(\log n)$ approximation algorithm for the Steiner ratio problem.

Since sparsest cut is a special case of the Steiner ratio problem, the $\Omega(\log n)$ integrality gap for the sparsest cut linear program [16] also implies that this approximation guarantee is tight for any algorithm based on the linear program $(LP - ratio)$.

3.2 Approximating requirement cut

As mentioned, the algorithm for requirement cut involves greedily augmenting the partial solution in phases, until all the residual requirements are 0. The greedy step in phase k is as follows. For each connected component of the residual graph G^k, run the Steiner ratio algorithm (Section 3.1), and among all the solutions, pick the cut B_k of minimum cost-effectiveness. Lemma 3 along with the Steiner ratio approximation implies that $eff(\partial_k B_k) \leq 4\rho \cdot \frac{OPT^*}{\phi_k}$. Using a standard analysis based on the greedy algorithm for set cover (see eg., [13]), we obtain that the total cost of the solution (over all phases) is at most $4\rho(\ln \phi_1 + 1) \cdot OPT^*$, where ϕ_1 is the total residual requirement in the first phase of the algorithm. Since $\phi_1 \leq gR$, the cost of the greedy solution is $O(\log n \cdot \log(gR)) \cdot OPT^*$. Clearly, the number of phases is at most gR, which gives a polynomial time algorithm.

Theorem 4 *The greedy algorithm is an $O(\log n \cdot \log(gR))$ approximation algorithm for the requirement cut problem on graphs.*

Note that when restricted to trees, the greedy step is trivial: cuts are just edges, and we can enumerate all of them. So the greedy algorithm is actually an $O(\log(gR))$ approximation for requirement cut on trees.

4 Conclusions

It would be interesting to obtain improved approximation guarantees for the requirement cut problem, even in special cases such as planar graphs. One approach could be to improve the approximation guarantee for the Steiner ratio problem. Arora et al. [2] used a semidefinite programming relaxation for sparsest cut (which is a special case of the Steiner ratio problem) to obtain an $O(\sqrt{\log n})$ approximation algorithm. However those techniques do not seem to apply directly to the Steiner ratio problem, and it would be interesting to see if a suitable SDP relaxation yields a stronger bound.

References

[1] Sanjeev Arora, James R. Lee, and Assaf Naor. Euclidean distortion and the Sparsest Cut. *Journal of the American Mathematical Society*, 21:1–21, 2008.

[2] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the 36th annual ACM symposium on Theory of computing*, pages 222–231, 2004.

[3] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ Approximate Min-Cut Max-Flow Theorem and Approximation Algorithm. *SIAM Journal on Computing*, 27(1):291–301, 1998.

[4] A. Avidor and M. Langberg. The Multi-Multiway cut problem. *Theoretical Computer Science*, 377(1-3):35–42, 2007.

[5] Gruia Calinescu, Howard J. Karloff, and Yuval Rabani. An Improved Approximation Algorithm for Multiway Cut. *Journal of Computer and System Sciences*, 60(3):564–574, 2000.

[6] Chandra Chekuri and Sudipto Guha. The Steiner k-cut problem. *SIAM Journal on Discrete Mathematics*, 20(1):261–271, 2006.

[7] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[8] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[9] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[10] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate Max-flow Min-(multi)cut Theorems and Their Applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[11] D.R. Karger, P.N. Klein, C. Stein, M. Thorup, and N.E. Young. Rounding algorithms for a geometric embedding for minimum multiway cut. *Mathematics of Operations Research*, 29(3):436–461, 2004.

[12] Philip N. Klein, Serge A. Plotkin, Satish Rao, and Eva Tardos. Approximation Algorithms for Steiner and Directed Multicuts. *Journal of Algorithms*, 22(2):241–269, 1997.

[13] Philip N. Klein and R. Ravi. A Nearly Best-Possible Approximation Algorithm for Node-Weighted Steiner Trees. *Journal of Algorithms*, 19(1):104–115, 1995.

[14] P.N. Klein, S. Rao, A. Agarwal, and R. Ravi. An approximate max-flow min-cut relation for multicommodity flow, with applications. *Combinatorica*, 15:187–202.

[15] Goran Konjevod, R. Ravi, and Aravind Srinivasan. Approximation algorithms for the covering Steiner problem. *Random Structures and Algorithms*, 20(3):465–482, 2002.

[16] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999.

[17] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 2005.

[18] Jr. L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8, 1956.

[19] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[20] Viswanath Nagarajan and R Ravi. Approximation algorithms for requirement cut on graphs. In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 209–220, 2005.

[21] Joseph Naor and Yuval Rabani. Tree packing and approximating k-cuts. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 26–27, 2001.

[22] R. Ravi and Amitabh Sinha. Approximating k-cuts via network strength. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 621–622, 2002.

[23] Huzur Saran and Vijay V. Vazirani. Finding k Cuts within Twice the Optimal. *SIAM Journal on Computing*, 24(1):101–108, 1995.